UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/661,982	09/12/2003	Cary Lee Bates	ROC920000051.D1	9327

46797 7590 10/05/2007

IBM CORPORATION, INTELLECTUAL PROPERTY LAW
DEPT 917, BLDG. 006-1
3605 HIGHWAY 52 NORTH
ROCHESTER, MN 55901-7829

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
10/05/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

D

Office Action Summary	Application No.	Applicant(s)	
	10/661,982	BATES ET AL.	
	Examiner	**Art Unit**	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 September 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date *12/16/2003*.
4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-22 are pending in this application and presented for examination.

Claim Rejections – 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

2. Claims 14-21 are rejected under 35 U.S.C 101 because the claims are directed to non-statutory subject matter.

3. **As to claims 14** and **16**, the claims recite a "computer-readable medium containing a program" to include transmission type (signaling), [0043], in the specifications; the claims are directed to a computer program product encoding a computer program. However, Applicant defines "computer-readable medium" to include "a computer data signal embodied in a carrier wave". Signals do not fall within any class of statutory subject matter, and thus the claim is not limited to statutory subject matter. Please see Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility (1300 OG 142), Annex IV, Section (C) for details.

4. **As to claim 15**, it does not cure the deficiency of base claim 14, and also are rejected under 35 U.S.C. 101 as set forth above.

5. **As to claims 17-21**, they do not cure the deficiency of base claim 16, and also are rejected under 35 U.S.C. 101 as set forth above.

Claim Rejections – 35 USC § 102(e)

The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness rejections set forth in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-22 are rejected under 35 U.S.C. 102(e) as being anticipated by Sato et al. (Pat. No. US 6,467,075 B1) (hereinafter 'Sato')

7. **As to claim 1**, Sato discloses a method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

- allowing a user to establish a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated,

the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized; Col. 14, Lines 46-53 – the techniques presented here could also be targeted to a software implementation);

- allowing the code to execute; upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated; and
- if so, notifying the user (e.g., Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators corresponding to each memory segment …; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

8. **As to claim 2** (incorporating the rejection in claim 1), Sato discloses the method wherein notifying the user comprises halting execution of the code (e.g., Col. 3, Lines 55-64).

9. **As to claim 3** (incorporating the rejection in claim 1), Sato discloses the method wherein notifying the user comprises halting execution of the code and displaying a status message to the user (e.g., Col. 3, Lines 55-64).

10. **As to claim 4** (incorporating the rejection in claim 1), Sato discloses the method if the relationship is not violated, freeing the memory space (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized).

11. **As to claim 5** (incorporating the rejection in claim 1), Sato discloses the method wherein determining whether the relationship is violated comprises determining that the memory space was allocated by an allocator different from the one or more memory allocators (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized).

12. **As to claim 6**, Sato discloses a method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

- establishing a relationship between a user-selected memory deallocator and a user-selected memory allocator, wherein the relationship requires that memory space freed by the user-selected deallocator have been allocated by the user-selected allocator (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized; Col. 14, Lines 46-53 – the techniques presented here could also be targeted to a software implementation);
- allowing the code to execute;
- upon a call to the user-selected deallocator to free a memory space, determining whether the memory space was allocated by the user-selected allocator; and
- if so, notifying the user that the relationship is violated (e.g., Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators corresponding to each memory

segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

13. **As to claim 7** (incorporating the rejection in claim 6), please refer to claim **3** as set forth above accordingly.

14. **As to claim 8**, Sato discloses a method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators (e.g., Fig. 10 – interface of the allocator block implementing malloc and free functions; Col. 4, Lines 3-24; Col. 14, Lines 46-62 – the techniques presented here could also be targeted to a software implementation), comprising:

- setting an upper limit on the amount of memory space an allocator can allocate during execution of the code, wherein the upper limit is specific to the allocator;
- during execution of the code, tracking the amount of memory space allocated by the allocator; and
- when the amount of memory space allocated exceeds the limit, notifying a user (e.g., Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the

size of each memory segment; the tool instantiates then the allocators corresponding to each memory segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated; Col. 14, Lines 46-53 – the techniques presented here could also be targeted to a software implementation).

15. **As to claim 9** (incorporating the rejection in claim 8), Sato discloses the step of tracking comprises:

- determining whether the allocator is called to allocate memory and, if so, incrementing a counter; and
- determining whether a deallocator is called to deallocate memory allocated by the allocator and, if so, decrementing the counter (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized).

16. **As to claim 10** (incorporating the rejection in claim 8), Sato discloses the step of tracking comprises incrementing a counter in the event of memory allocation by the allocator and decrementing the counter in the event of memory deallocation of memory space allocated by the allocator (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized).

17. **As to claim 11** (incorporating the rejection in claim 8), Sato discloses notifying the user comprises halting execution of the code (e.g., Col. 3, Lines 55-64).

18. **As to claim 12** (incorporating the rejection in claim 8, Sato discloses wherein the upper limit is independent of other memory size limitations (e.g., Col. 9, Line 41 through Col. 10, Line 29 – e.g., allocate memory in *local_RAM* or allocate memory in *shared_RAM*; Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators corresponding to each memory segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced

by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

19. **As to claim 13** (incorporating the rejection in claim 8), Sato discloses wherein the upper limit is not a limit on a stack size (e.g., Col. 9, Line 41 through Col. 10, Line 29 – e.g., allocate memory in *local_RAM* or allocate memory in *shared_RAM*; Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators corresponding to each memory segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

20. **As to claim 14**, Sato discloses a computer readable medium containing a program which, when executed, performs an operation for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, the operation comprising:

- establishing a relationship between a user-selected memory deallocator and a user-selected memory allocator, wherein the relationship requires that memory space freed by the user-selected deallocator have been allocated by the user-selected allocator (e.g., Col. 17, Lines 59-67 – the idea is to have a counter for each dynamically-allocated location set; during the analysis, the counter is incremented each time an element of the corresponding location set is allocated; subsequently, each time an element of the location set is deallocated, the associated counter is decremented; this way, location sets allocated and not deallocated within these locations cannot be optimized; otherwise, they can be optimized; Col. 14, Lines 46-53 – the techniques presented here could also be targeted to a software implementation);
- allowing the code to execute;
- upon a call to the user-selected deallocator to free a memory space, determining whether the memory space was allocated by the user-selected allocator; and
- if so, notifying the user that the relationship is violated (e.g., Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators corresponding to each memory segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called

according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

21. **As to claim 15** (incorporating the rejection in claim 1), please refer to claim **3** as set forth above accordingly.

22. **As to claim 16,** Sato discloses a computer readable medium containing a program which, when executed, performs an operation for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators (e.g., Fig. 10 – interface of the allocator block implementing malloc and free functions; Col. 4, Lines 3-24; Col. 14, Lines 46-62 – the techniques presented here could also be targeted to a software implementation; Col. 14, Lines 46-53 – the techniques presented here could also be targeted to a software implementation), the operation comprising:

- setting an upper limit on the amount of memory space an allocator can allocate during execution of the code, wherein the upper limit is specific to the allocator;
- during execution of the code, tracking the amount of memory space allocated by the allocator; and
- when the amount of memory space allocated exceeds the limit, notifying a user (e.g., Col. 14, Line 64 through Col. 15, Line 29 – since the size of the dynamically allocated memory is a priori unknown at compile time, the designer also sets the size of each memory segment; the tool instantiates then the allocators

corresponding to each memory segment ...; for each memory segment, a different allocator is instantiated; each malloc mapped to this memory segment is then replaced by a call to the specific allocator; the inventors generate a branching statement in which the different allocators corresponding the different memory segments may be called according to the pointer's tag; the pointer's index is then sent to the allocator to indicate which block should be deallocated).

23. **As to claims 17-21**, please refer to claims **9-13** as set forth above accordingly.

Conclusion

24. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Séméria et al., "Resolution of Dynamic Memory Allocation and Pointers for the Behavioral Synthesis from C, January 2000, Computer Systems Laboratory, Stanford University, pp. 1-8
- Kolawa et al., "Method and System for Dynamically Detecting Leaked Memory Space in a Computer Program" (Pat. No. 5,842,019)
- Georg et al., "Dynamic Physical Memory Mapping and Management of Independent Programming Environments" (Pat. No. 4,511,964)

25. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-

1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (iN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW

September 25, 2007